An Informal Explanation of Simple Consensus (2/3)
by David Guaspari ATC-NY November 29, 2077

Rep1 receives a proposal for command x and broadcasts its vote.
Rep2 and Rep3 receive these votes and immediately reply with their votes for x, which Rep1 receives.
The vote queues in Quorum now look like this:
Rep1: [x;x;x] -- so Rep1 will broadcast a decision and terminate
Rep2: [x;x]
Rep3: [x;x]
Rep4: []

Continue. There are three votes in transit, all for x; before they arrive, Rep4 receives a proposal for command y and broadcasts its vote. Rep2 and Rep3 receive this vote.
The vote queues in Quorum now look like this:
Rep2: [x;x;y] -- will retry, proposing x
Rep3: [x;x;y] -- will retry, proposing x
Rep4: [y]

Since Rep2 and Rep3 have a majority of votes for x, they will send retry messages that propose x; and, since the remaining votes in transit are votes for x, Rep4 cannot send a retry proposing y. In this case, all the Reps will decide on x in the next round.

If the command to propose on retry were chosen nondeterministically, then the next round could come to consensus on y. So:

a) There must be some restriction on what a retry proposes. If there are two distinct retry proposals for the same round, then it's possible to come to consensus on either one of them.

b) Accordingly, the invariant we need is this: If, in any round, some Replica comes to consensus, then, in that round, there is only one command that can possibly be proposed by a retry. (So it follows that every other Replica that does not come to consensus on this round will on the next.)

Here's why (b) is true:

Suppose that one Replica sees 2f+1 votes for command x in a given round.
Consider the situation of any other Replica when its Quorum class makes a decision: It has receive 2f+1 votes; but, in this round, there can only be f votes for a command other than x; so a majority of its votes must be for x; so the only possible retries propose x.

Here, by the way, is a scenario that never comes to consensus:

Rep1 receives a proposal for x and Rep2 receives Rep1's vote for x.
Rep4 receives a proposal for y and Rep3 receives Rep4's vote for y.
Vote queues:
Rep1: [x]
Rep2: [x]
Rep3: [y]
Rep4: [y]

Rep2 replies to Rep1 and Rep3 to Rep4.
Vote queues:

Rep1: [x;x]
Rep2: [x]
Rep3: [y]
Rep4: [y;y]

Rep1 and Rep4 receive each other's votes.
Vote queues:
Rep1: [x;x;y] -- retry, proposing x
Rep2: [x]
Rep3: [y]
Rep4: [y;y;x] -- retry, proposing y

This could repeat indefinitely without ever coming to consensus.